Auto-Scaling Self-Hosted Agents in Azure Devops

**Create Dockerfile for Self Hosted Agent:**

**Install required Dependices for running pipelines JOBS**

FROM ubuntu:latest

RUN apt-get update

RUN apt-get upgrade -y

RUN apt-get install -y \

    apt-transport-https \

    apt-utils \

    ca-certificates \

    curl \

    git \

    iputils-ping \

    jq \

    lsb-release \

    software-properties-common

RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash

# Can be 'linux-x64', 'linux-arm64', 'linux-arm', 'rhel.6-x64'.

ENV TARGETARCH=linux-x64

RUN apt-get install gnupg wget -y

# install Java and Maven

RUN apt-get install openjdk-11-jdk maven -y

# install buildctl binary

RUN wget https://github.com/moby/buildkit/releases/download/v0.11.6/buildkit-v0.11.6.linux-amd64.tar.gz

RUN tar -xvzf \*.tar.gz

WORKDIR /azp

COPY start.sh .

RUN chmod +x start.sh

ENTRYPOINT ["./start.sh"]

**Script Start.sh:**

#!/bin/bash

set -e

if [ -z "$AZP\_URL" ]; then

  echo 1>&2 "error: missing AZP\_URL environment variable"

  exit 1

fi

if [ -z "$AZP\_TOKEN\_FILE" ]; then

  if [ -z "$AZP\_TOKEN" ]; then

    echo 1>&2 "error: missing AZP\_TOKEN environment variable"

    exit 1

  fi

  AZP\_TOKEN\_FILE=/azp/.token

  echo -n $AZP\_TOKEN > "$AZP\_TOKEN\_FILE"

fi

unset AZP\_TOKEN

if [ -n "$AZP\_WORK" ]; then

  mkdir -p "$AZP\_WORK"

fi

export AGENT\_ALLOW\_RUNASROOT="1"

cleanup() {

  if [ -e config.sh ]; then

    print\_header "Cleanup. Removing Azure Pipelines agent..."

    # If the agent has some running jobs, the configuration removal process will fail.

    # So, give it some time to finish the job.

    while true; do

      ./config.sh remove --unattended --auth PAT --token $(cat "$AZP\_TOKEN\_FILE") && break

      echo "Retrying in 30 seconds..."

      sleep 30

    done

  fi

}

print\_header() {

  lightcyan='\033[1;36m'

  nocolor='\033[0m'

  echo -e "${lightcyan}$1${nocolor}"

}

# Let the agent ignore the token env variables

export VSO\_AGENT\_IGNORE=AZP\_TOKEN,AZP\_TOKEN\_FILE

print\_header "1. Determining matching Azure Pipelines agent..."

AZP\_AGENT\_PACKAGES=$(curl -LsS \

    -u user:$(cat "$AZP\_TOKEN\_FILE") \

    -H 'Accept:application/json;' \

    "$AZP\_URL/\_apis/distributedtask/packages/agent?platform=$TARGETARCH&top=1")

AZP\_AGENT\_PACKAGE\_LATEST\_URL=$(echo "$AZP\_AGENT\_PACKAGES" | jq -r '.value[0].downloadUrl')

if [ -z "$AZP\_AGENT\_PACKAGE\_LATEST\_URL" -o "$AZP\_AGENT\_PACKAGE\_LATEST\_URL" == "null" ]; then

  echo 1>&2 "error: could not determine a matching Azure Pipelines agent"

  echo 1>&2 "check that account '$AZP\_URL' is correct and the token is valid for that account"

  exit 1

fi

print\_header "2. Downloading and extracting Azure Pipelines agent..."

curl -LsS $AZP\_AGENT\_PACKAGE\_LATEST\_URL | tar -xz & wait $!

source ./env.sh

trap 'cleanup; exit 0' EXIT

trap 'cleanup; exit 130' INT

trap 'cleanup; exit 143' TERM

print\_header "3. Configuring Azure Pipelines agent..."

./config.sh --unattended \

  --agent "${AZP\_AGENT\_NAME:-$(hostname)}" \

  --url "$AZP\_URL" \

  --auth PAT \

  --token $(cat "$AZP\_TOKEN\_FILE") \

  --pool "${AZP\_POOL:-Default}" \

  --work "${AZP\_WORK:-\_work}" \

  --replace \

  --acceptTeeEula & wait $!

print\_header "4. Running Azure Pipelines agent..."

trap 'cleanup; exit 0' EXIT

trap 'cleanup; exit 130' INT

trap 'cleanup; exit 143' TERM

chmod +x ./run.sh

# To be aware of TERM and INT signals call run.sh

# Running it with the --once flag at the end will shut down the agent after the build is executed

./run.sh "$@" & wait $!

1. **Create docker Image using above Dockerfile**

**Docker build -t regitsryname.azurecr.io/imagename:tag .**

1. **Generate PAT TOKEN and provide required access**

Create secret for TOKEN:

apiVersion: v1

kind: Secret

metadata:

  name: azdevops

data:

  AZP\_TOKEN: <base64 encoded PAT token>

**Deployment file for AzDevOps Agents:**

**Make sure the agent pool with name provided below should be created manually at Azure DevOps console**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: azdevops-deployment

  labels:

    app: azdevops-agent

spec:

  replicas: 1

  selector:

    matchLabels:

      app: azdevops-agent

  template:

    metadata:

      labels:

        app: azdevops-agent

    spec:

      containers:

      - name: azdevops-agent

        image: icsitalentdev.azurecr.io/agent-test

        env:

          - name: AZP\_URL

            value: "https://dev.azure.com/vinaytalla20"

          - name: AZP\_POOL

            value: "kubernetes"

          - name: AZP\_TOKEN

            valueFrom:

              secretKeyRef:

                name: azdevops

                key: AZP\_TOKEN

**Install keda**

helm repo add kedacore <https://kedacore.github.io/charts>

helm repo update

kubectl create namespace keda

helm install keda kedacore/keda --namespace keda

**Uninstall**

If you want to remove KEDA from a cluster, you first need to remove any ScaledObjects and ScaledJobs that you have created. Once that is done, the Helm chart can be uninstalled:

kubectl delete $(kubectl get scaledobjects.keda.sh,scaledjobs.keda.sh -A \

-o *jsonpath*='{"-n "}{.items[\*].metadata.namespace}{" "}{.items[\*].kind}{"/"}{.items[\*].metadata.name}{"\n"}')

helm uninstall keda -n keda

Note: if you uninstall the Helm chart without first deleting any ScaledObject or ScaledJob resources you have created, they will become orphaned. In this situation, you will need to patch the resources to remove their finalizers. Once this is done, they should automatically be removed:

for i in $(kubectl get scaledobjects -A \

-o *jsonpath*='{"-n "}{.items[\*].metadata.namespace}{" "}{.items[\*].kind}{"/"}{.items[\*].metadata.name}{"\n"}');

do kubectl patch *$i* -p '{"metadata":{"finalizers":null}}' --type=merge

done

for i in $(kubectl get scaledjobs -A \

-o *jsonpath*='{"-n "}{.items[\*].metadata.namespace}{" "}{.items[\*].kind}{"/"}{.items[\*].metadata.name}{"\n"}');

do kubectl patch *$i* -p '{"metadata":{"finalizers":null}}' --type=merge

done

After the deployment is created you need to create the ScaledObject in order for KEDA to start scaling the deployment

To scale based on the queue length of an Azure Pipelines agent pool, you can use the azure-pipelines

**Create Secret for KEDA to access Azure DevOps**

apiVersion: v1

kind: Secret

metadata:

name: pipeline-auth

data:

personalAccessToken: <base64 encoded PAT>

**Create Trigger Authentication:**

apiVersion: keda.sh/v1alpha1

kind: TriggerAuthentication

metadata:

name: pipeline-trigger-auth

spec:

secretTargetRef:

- parameter: personalAccessToken

name: pipeline-auth

key: personalAccessToken

**Create Scaled Object :**

1. **Modify the pool id of the one you want to use, to get pool ID use REST API URL:**

[https://dev.azure.com/{organization\_name}/\_apis/distributedtask/pools?api-version=6.1-preview.1](https://dev.azure.com/%7borganization_name%7d/_apis/distributedtask/pools?api-version=6.1-preview.1)

1. **Hit above URL in browser, update organization name with our org\_name**
2. **Also can change number of Replicas to required number**

apiVersion: keda.sh/v1alpha1

kind: ScaledObject

metadata:

name: azure-pipelines-scaledobject

spec:

scaleTargetRef:

name: azdevops-deployment

minReplicaCount: 1

maxReplicaCount: 5

triggers:

- type: azure-pipelines

metadata:

poolID: "1"

organizationURLFromEnv: "AZP\_URL"

authenticationRef:

name: pipeline-trigger-auth

**For Reference:**

To run jobs with specific agent pool:

pool:

  name: poolname

**Used buildkitd instead of docker in agent pod, because the Kubernetes cluster runs with containerd**

**To use buildkitd as replacement for docker:**

**Follow below steps:**

1. **Setup buildkitd Deamon in Kubernetes cluster, with below YAML**

apiVersion: apps/v1

kind: Deployment

metadata:

  labels:

    app: buildkitd

  name: buildkitd

spec:

  replicas: 1

  selector:

    matchLabels:

      app: buildkitd

  template:

    metadata:

      labels:

        app: buildkitd

    spec:

      containers:

        - name: buildkitd

          image: moby/buildkit:master

          args:

            - --addr

            - unix:///run/buildkit/buildkitd.sock

            - --addr

            - tcp://0.0.0.0:1234

          # the probe below will only work after Release v0.6.3

          readinessProbe:

            exec:

              command:

                - buildctl

                - debug

                - workers

            initialDelaySeconds: 5

            periodSeconds: 30

          # the probe below will only work after Release v0.6.3

          livenessProbe:

            exec:

              command:

                - buildctl

                - debug

                - workers

            initialDelaySeconds: 5

            periodSeconds: 30

          securityContext:

            privileged: true

          ports:

            - containerPort: 1234

---

apiVersion: v1

kind: Service

metadata:

  labels:

    app: buildkitd

  name: buildkitd

spec:

  ports:

    - port: 1234

      protocol: TCP

  selector:

    app: buildkitd

1. **Using Buitctl to build images and push them to Private Repository:**
2. **Download Buildkitd binary:**

* source url: <https://github.com/moby/buildkit/releases/download/v0.11.6/buildkit-v0.11.6.linux-amd64.tar.gz>
* extract the tar ball

1. **Setup Private Registry Credentials with using Docker login:**

BASE64\_AUTH=`echo -n "$CI\_REGISTRY\_USER:$CI\_REGISTRY\_PASSWORD" | base64 -w0`

mkdir -p ~/.docker

echo "{\"auths\": {\"$CI\_REGISTRY\": {\"auth\": \"$BASE64\_AUTH\"}}}" > ~/.docker/config.json

1. **To build and push Images:**

**/home/vinay/bin/buildctl --addr tcp://buildkitd.default.svc.cluster.local:1234 build --frontend=dockerfile.v0 --local context=. --local dockerfile=. --output type=image,name=icsitalentdev.azurecr.io/test-push,push=true**